

EE31806210505
June 8, 2009

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Docket No. AUS990940US1

Assistant Commissioner for Patents
Washington, D.C. 20231

Sir:

Transmitted herewith for filing is the patent application of Inventor(s):

RICHARD LOUIS ARNDT

For: **HYPERVERSOR AS A SET OF SERVICES**

Enclosed are also:

<u>X</u>	21	Pages of Specification including an Abstract
<u>X</u>	7	Pages of Claims
<u>X</u>	6	Sheet(s) of Drawings
<u>X</u>		A Declaration and Power of Attorney
<u>X</u>		Form PTO 1595 and assignment of the invention to IBM Corporation

CLAIMS AS FILED

FOR	Number Filed		Number Extra		Rate		Basic Fee (\$690)
Total Claims	28	-20 =	8	X	\$ 18	=	\$ 144.00
Independent Claims	7	-3 =	4	X	\$ 78	=	\$ 312.00
Multiple Dependent Claims	0			X	\$260	=	\$ 0.00
Total Filing Fee =							\$1,146.00

X Please charge \$ 1,146.00 to IBM Corporation, Deposit Account No. 09-0447.

X The Commissioner is hereby authorized to charge payment of the following fees associated with the communication or credit any over payment to IBM Corporation, Deposit Account No. 09-0447. A duplicate copy of this sheet is enclosed.

X Any additional filing fees required under 37CFR § 1.16.

X Any patent application processing fees under 37CFR § 1.17.

Respectfully,

Mark E. McBurney

Reg. No. 33,114

Intellectual Property Law Dept.

IBM Corporation

11400 Burnet Road 4054

Austin, Texas 75758

Telephone: (512) 823-1003

5

BACKGROUND OF THE INVENTION

10

15

20

2. Description of Related Art:

30

Docket No. AUS990940US1

management area, regions of system memory, and I/O adapter bus slots. The partition's resources are represented by its own open firmware device tree to the OS image.

5 Each distinct OS or image of an OS running within the platform are protected from each other, such that software errors on one logical partition cannot affect the correct operation of any of the other partitions. This is provided by allocating a disjoint set of platform
10 resources to be directly managed by each OS image and by providing mechanisms for ensuring that the various images can not control any resources that have not been allocated to it. Furthermore, software errors in the control of an OS's allocated resources are prevented from
15 affecting the resources of any other image. Thus, each image of the OS (or each different OS) directly controls a distinct set of allocable resources within the platform.

20 A significant problem with logically partitioned platform has been "when" to run trusted platform firmware to create and enforce the partitioning of the platform's resources. In general, there have been two answers. One solution is to run the firmware prior to starting the OS image, at which time special hardware is set up to
25 restrict the access that the OS image can make. The other solution is to run the firmware when certain privileged instructions that may change the environment are trapped (i.e. prevented from executing). Both of these answers have drawbacks. The special hardware
30 solution results in overly constraining the OS image to run in a fixed set of resources that can be managed by

the hardware. The trapping on privileged instruction solution tends to create a significant performance penalty since most of the time the trapped instructions were being used for other purposes than to change the OS image's environment. Therefore, a method of creating and enforcing the partitioning of a platform's resources that allows for the significant flexibility in the assignment of resources of the instruction trap approach, and that also maintains a level of performance more closely associated with the hardware approaches is desirable.

Docket No. AUS990940US1

SUMMARY OF THE INVENTION

The present invention provides a method, apparatus, and system for preventing each of a plurality of operating system within a logically partitioned data processing system from interfering with the operation of the other operating systems. In one embodiment, a logically partitioned data processing system includes a plurality of logical partitions; a plurality of operating systems, a plurality of assignable resources, at least one non-assignable resource, and a hypervisor. Each of the plurality of operating systems is assigned to a separate one of the plurality of logical partitions and each of the plurality of assignable resources is assigned to one of the plurality of logical partitions. The hypervisor provides a set of services to each of the plurality of logical partitions, wherein these services safely perform modifications to non-assignable processing system resources in response to operating system requests without allowing direct access to the non-assignable resources by the operating system image. Thus, each operating system is prevented from modifying the non-assignable resource in such a way that interferes with the operation of other ones of the plurality of operating systems.

Docket No. AUS990940US1

BRIEF DESCRIPTION OF THE DRAWINGS

The novel features believed characteristic of the invention are set forth in the appended claims. The invention itself, however, as well as, a preferred mode of use, further objectives and advantages thereof, will best be understood by reference to the following detailed description of an illustrative embodiment when read in conjunction with the accompanying drawings, wherein:

Figure 1 depicts a pictorial representation of a distributed data processing system in which the present invention may be implemented;

Figure 2, a block diagram of a data processing system in accordance with the present invention is illustrated;

Figure 3 depicts a block diagram of a data processing system, which may be implemented as a logically partitioned server, in accordance with the present invention;

Figure 4 depicts a block diagram of a logically partitioned platform in which the present invention may be implemented;

Figure 5 depicts a flowchart illustrating an exemplary process for preventing each of multiple OS images within a logically partitioned platform from interfering with other OS images in accordance with the present invention; and

Figure 6 depicts a high level flowchart illustrating an exemplary process within an operating system image for requesting operations to a logically partitioned platform

Docket No. AUS990940US1

in accordance with the present invention.

[illegible]

Docket No. AUS990940US1

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

With reference now to the figures, and in particular
5 with reference to **Figure 1**, a pictorial representation of
a distributed data processing system is depicted in which
the present invention may be implemented.

Distributed data processing system **100** is a network
of computers in which the present invention may be
10 implemented. Distributed data processing system **100**
contains network **102**, which is the medium used to provide
communications links between various devices and
computers connected within distributed data processing
system **100**. Network **102** may include permanent
15 connections, such as wire or fiber optic cables, or
temporary connections made through telephone connections.

In the depicted example, server **104** is connected to
hardware system console **150**. Server **104** is also
connected to network **102**, along with storage unit **106**.
20 In addition, clients **108**, **110** and **112** are also connected
to network **102**. These clients, **108**, **110** and **112**, may be,
for example, personal computers or network computers.
For purposes of this application, a network computer is
any computer coupled to a network that receives a program
25 or other application from another computer coupled to the
network. In the depicted example, server **104** is a
logically partitioned platform and provides data, such as
boot files, operating system images and applications, to
clients **108-112**. Hardware system console **150** may be a
30 laptop computer and is used to display messages to an

Docket No. AUS990940US1

operator from each operating system image running on
server **104**, as well as to send input information,
received from the operator, to server **104**. Clients **108**,
110 and **112** are clients to server **104**. Distributed data
5 processing system **100** may include additional servers,
clients, and other devices not shown. Distributed data
processing system **100** also includes printers **114**, **116** and
118. A client, such as client **110**, may print directly to
printer **114**. Clients such as client **108** and client **112**
10 do not have directly attached printers. These clients
may print to printer **116**, which is attached to server
104, or to printer **118**, which is a network printer that
does not require connection to a computer for printing
documents. Client **110**, alternatively, may print to
15 printer **116** or printer **118**, depending on the printer type
and the document requirements.

In the depicted example, distributed data processing
system **100** is the Internet, with network **102** representing
a worldwide collection of networks and gateways that use
20 the TCP/IP suite of protocols to communicate with one
another. At the heart of the Internet is a backbone of
high-speed data communication lines between major nodes
or host computers consisting of thousands of commercial,
government, education, and other computer systems that
25 route data and messages. Of course, distributed data
processing system **100** also may be implemented as a number
of different types of networks such as, for example, an
intranet or a local area network.

Figure 1 is intended as an example and not as an
30 architectural limitation for the processes of the present

003090" E9952560

Docket No. AUS990940US1

invention.

With reference now to **Figure 2**, a block diagram of a data processing system in accordance with the present invention is illustrated. Data processing system **200** is an example of a hardware system console, such as hardware system console **150** depicted in **Figure 1**. Data processing system **200** employs a peripheral component interconnect (PCI) local bus architecture. Although the depicted example employs a PCI bus, other bus architectures, such as Micro Channel and ISA, may be used. Processor **202** and main memory **204** are connected to PCI local bus **206** through PCI bridge **208**. PCI bridge **208** may also include an integrated memory controller and cache memory for processor **202**. Additional connections to PCI local bus **206** may be made through direct component interconnection or through add-in boards. In the depicted example, local area network (LAN) adapter **210**, SCSI host bus adapter **212**, and expansion bus interface **214** are connected to PCI local bus **206** by a direct component connection. In contrast, audio adapter **216**, graphics adapter **218**, and audio/video adapter (A/V) **219** are connected to PCI local bus **206** by add-in boards inserted into expansion slots. Expansion bus interface **214** provides a connection for a keyboard and mouse adapter **220**, modem **222**, and additional memory **224**. In the depicted example, SCSI host bus adapter **212** provides a connection for hard disk drive **226**, tape drive **228**, CD-ROM drive **230**, and digital video disc read only memory drive (DVD-ROM) **232**. Typical PCI local bus implementations will support three or four PCI expansion slots or add-in connectors.

Docket No. AUS990940US1

003090" 636550

An operating system runs on processor **202** and is used to coordinate and provide control of various components within data processing system **200** in **Figure 2**. The operating system may be a commercially available
5 operating system, such as OS/2, which is available from International Business Machines Corporation. "OS/2" is a trademark of International Business Machines Corporation. An object oriented programming system, such as Java, may run in conjunction with the operating system, providing
10 calls to the operating system from Java programs or applications executing on data processing system **200**. Instructions for the operating system, the object-oriented operating system, and applications or programs are located on a storage device, such as hard
15 disk drive **226**, and may be loaded into main memory **204** for execution by processor **202**.

Those of ordinary skill in the art will appreciate that the hardware in **Figure 2** may vary depending on the implementation. For example, other peripheral devices,
20 such as optical disk drives and the like, may be used in addition to or in place of the hardware depicted in **Figure 2**. The depicted example is not meant to imply architectural limitations with respect to the present invention. For example, the processes of the present
25 invention may be applied to multiprocessor data processing systems.

With reference now to **Figure 3**, a block diagram of a data processing system, which may be implemented as a logically partitioned server, such as server **104** in
30 **Figure 1**, is depicted in accordance with the present invention. Data processing system **300** may be a symmetric

multiprocessor (SMP) system including a plurality of processors **301**, **302**, **303**, and **304** connected to system bus **306**. For example, data processing system **300** may be an IBM RS/6000, a product of International Business Machines Corporation in Armonk, New York. Alternatively, a single processor system may be employed. Also connected to system bus **306** is memory controller/cache **308**, which provides an interface to a plurality of local memories **360-363**. I/O bus bridge **310** is connected to system bus **306** and provides an interface to I/O bus **312**. Memory controller/cache **308** and I/O bus bridge **310** may be integrated as depicted.

Data processing system **300** is a logically partitioned data processing system. Thus, data processing system **300** may have multiple heterogeneous operating systems (or multiple instances of a single operating system) running simultaneously. Each of these multiple operating systems may have any number of software programs executing within in it. Data processing system **300** is logically partitioned such that different I/O adapters **320-321**, **328-329**, **336-337**, and **346-347** may be assigned to different logical partitions.

Thus, for example, suppose data processing system 300 is divided into three logical partitions, P1, P2, and P3. Each of I/O adapters 320-321, 328-329, and 336-337, each of processors 301-304, and each of local memories 360-364 is assigned to one of the three partitions. For example, processor 301, memory 360, and I/O adapters 320, 328, and 329 may be assigned to logical partition P1; processors 302-303, memory 361, and I/O adapters 321 and

Docket No. AUS990940US1

337 may be assigned to partition P2; and processor **304**, memories **362-363**, and I/O adapters **336** and **346-347** may be assigned to logical partition P3.

Each operating system executing within data processing system **300** is assigned to a different logical partition. Thus, each operating system executing within data processing system **300** may access only those I/O units that are within its logical partition. Thus, for example, one instance of the Advanced Interactive Executive (AIX) operating system may be executing within partition P1, a second instance (image) of the AIX operating system may be executing within partition P2, and a Windows 2000™ operating system may be operating within logical partition P1. Windows 2000 is a product and trademark of Microsoft Corporation of Redmond, Washington.

Peripheral component interconnect (PCI) Host bridge **314** connected to I/O bus **312** provides an interface to PCI local bus **315**. A number of Terminal Bridges **316-317** may be connected to PCI bus **315**. Typical PCI bus implementations will support four Terminal Bridges for providing expansion slots or add-in connectors. Each of Terminal Bridges **316-317** is connected to a PCI/I/O Adapter **320-321** through a PCI Bus **318-319**. Each I/O Adapter **320-321** provides an interface between data processing system **300** and input/output devices such as, for example, other network computers, which are clients to server **300**. Only a single I/O adapter **320-321** may be connected to each terminal bridge **316-317**. Each of terminal bridges **316-317** is configured to prevent the

Docket No. AUS990940US1

propagation of errors up into the PCI Host Bridge **314** and into higher levels of data processing system **300**. By doing so, an error received by any of terminal bridges **316-317** is isolated from the shared buses **315** and **312** of the other I/O adapters **321**, **328-329**, and **336-337** that may be in different partitions. Therefore, an error occurring within an I/O device in one partition is not "seen" by the operating system of another partition. Thus, the integrity of the operating system in one partition is not effected by an error occurring in another logical partition. Without such isolation of errors, an error occurring within an I/O device of one partition may cause the operating systems or application programs of another partition to cease to operate or to cease to operate correctly.

Additional PCI host bridges **322**, **330**, and **340** provide interfaces for additional PCI buses **323**, **331**, and **341**. Each of additional PCI buses **323**, **331**, and **341** are connected to a plurality of terminal bridges **324-325**, **332-333**, and **342-343** which are each connected to a PCI I/O adapter **328-329**, **336-337**, and **346-347** by a PCI bus **326-327**, **334-335**, and **344-345**. Thus, additional I/O devices, such as, for example, modems or network adapters may be supported through each of PCI I/O adapters **328-329**, **336-337**, and **346-347**. In this manner, server **300** allows connections to multiple network computers. A memory mapped graphics adapter **348** and hard disk **350** may also be connected to I/O bus **312** as depicted, either directly or indirectly. Hard disk **350** may be logically partitioned between various partitions without the need

for additional hard disks. However, additional hard disks may be utilized if desired.

Those of ordinary skill in the art will appreciate that the hardware depicted in **Figure 3** may vary. For example, other peripheral devices, such as optical disk drives and the like, also may be used in addition to or in place of the hardware depicted. The depicted example is not meant to imply architectural limitations with respect to the present invention.

With reference now to **Figure 4**, a block diagram of an exemplary logically partitioned platform is depicted in which the present invention may be implemented. The hardware in logically partitioned platform **400** may be implemented as, for example, server **300** in **Figure 3**.

Logically partitioned platform **400** includes partitioned hardware **430**, hypervisor **410**, and operating systems **402-408**. Operating systems **402-408** may be multiple copies of a single operating system or multiple heterogeneous operating systems simultaneously run on platform **400**.

Partitioned hardware **430** includes a plurality of processors **432-438**, a plurality of system memory units **440-446**, a plurality of input/output (I/O) adapters **448-462**, and a storage unit **470**. Each of the processors **432-438**, memory units **440-446**, and I/O adapters **448-462** may be assigned to one of multiple partitions within logically partitioned platform **400**, each of which corresponds to one of operating systems **402-408**.

Hypervisor **410**, implemented as firmware, creates and enforces the partitioning of logically partitioned

Docket No. AUS990940US1

platform **400**. Firmware is "hard software" stored in a memory chip that holds its content without electrical power, such as, for example, read-only memory (ROM), programmable ROM (PROM), erasable programmable ROM (EPROM), electrically erasable programmable ROM (EEPROM), and non-volatile random access memory (non-volatile RAM).

Hypervisor **410** provides a set of firmware services to each of OS images **402-408** that safely modify the environment of each of OS images **402-408** such that interference between various ones of OS images **402-408** is prevented. Modification of the processor mode of each of processors **432-438** and address translation facilities is precluded by a hardware mode that is only deactivated when one of these hypervisor **410** services are run. Thus, the hypervisor **410** provides a small library of services that provide the richness of flexibility that hardware alone cannot provide without significantly adding to the performance overhead, since the services are firmware clones of those that the OS images would use on a standard non-partitioned platform.

To aid in understanding the present invention, consider the following example of an operating system image requesting to change an entry in a page frame table both in the context of the prior art and in the context of the present invention. A page frame table provides a translation between the logical address assigned to a system resource by an operating system image and the actual physical address of the resource within the data processing system. Each of these resources is assigned to one of OS images **402-408**. Thus, each of OS images **402-408** must be prevented from changing an entry in the

Docket No. AUS990940US1

page frame table corresponding to a resource assigned to a different one of OS images **402-408**.

In the prior art firmware trapping method, the firmware protected every location within the page frame table. Every write or read instruction to one of these locations within the page frame table by an operating system image was trapped by the firmware. The firmware then decoded the instruction to determine if the location that the operating system image was attempting to write to or read from corresponded to a resource assigned to that operating system image. If the location did correspond to a resource assigned to that particular operating system image, then the instruction was executed. If the location did not correspond to a resource assigned to that particular operating system image, then the instruction was not executed. Such a procedure requires an enormous amount of overhead to analyze every one of these questionably safe instructions, most of which are safe.

In an embodiment of the present invention, the operating system images are prohibited from writing to the page frame table entirely. However, the functional equivalent of changing an entry in the page frame table is provided by hypervisor **410**. Hypervisor **410** determines, in response to a function call from one of OS images **402-408**, what the one of OS images **402-408** wishes to change, such as, for example, translating a page from its own memory to an allocated resource, and creates a separate translation table to translate the memory page to the allocated resource. Such a high level translation requires significantly less overhead than tracking every

Docket No. AUS990940US1

instruction from the OS image. Each OS system image
402-408 running on logical partitioned platform **400** is
modified such that rather than request to perform a
questionably safe operation, the OS image **402-408** makes a
5 request to the hypervisor **410** for the hypervisor **410** to
provide the functional equivalent.

It should be noted that translating a page from
memory to an allocated resource is but one example of a
"questionably" safe operation. Other "questionably" safe
10 operations to non-assignable resources may be performed
by hypervisor **410** to insure that such operations are
performed such that interaction with other partitions
within the system is prevented.

Hypervisor **410** also may provide the OS images
15 **402-408** running in multiple logical partitions each a
virtual copy of a console and operator panel. The
interface to the console is changed from an asynchronous
teletype port device driver, as in the prior art, to a
set of hypervisor firmware calls that emulate a port
20 device driver. The hypervisor **410** encapsulates the data
from the various OS images onto a message stream that is
transferred to a computer **480**, known as a hardware system
console.

Hardware system console **480** is connected directly to
25 logically partitioned platform **400** as illustrated in
Figure 4, or may be connected to logically partitioned
platform through a network, such as, for example, network
102 in **Figure 1**. Hardware system console **480** may be, for
example, a desktop or laptop computer, and may be
30 implemented as data processing system **200** in **Figure 2**.

Docket No. AUS990940US1

Hardware system console **480** decodes the message stream and displays the information from the various OS images **402-408** in separate windows, at least one per OS image. Similarly, keyboard input information from the operator
5 is packaged by the hardware system console, sent to logically partitioned platform **400** where it is decoded and delivered to the appropriate OS image via the hypervisor **410** emulated port device driver associated with the then active window on the hardware system
10 console **480**.

Those of ordinary skill in the art will appreciate that the hardware and software depicted in **Figure 4** may vary. For example, more or fewer processors and/or more or fewer operating system images may be used than those
15 depicted in **Figure 4**. The depicted example is not meant to imply architectural limitations with respect to the present invention.

With reference now to **Figure 5**, a flowchart illustrating an exemplary process for preventing each of
20 multiple OS images within a logically partitioned platform from interfering with other OS images is depicted in accordance with the present invention. The hypervisor receives a request from an operating system image to perform a potentially hazardous operation (step
25 **502**). The hypervisor then determines if the result of the request would allow the OS direct access to platform resources outside of those allocated to its partition (step **504**). For example, the OS may wish to map an existing partition resource to a new memory address. The
30 hypervisor knows this request is safe. If the request would not result in direct OS access to an unassigned

Docket No. AUS990940US1

resource, then the hypervisor performs the OS requested function (step **506**). Continuing the example, the hypervisor then creates a translation table entry to achieve the OS image's desired result. Thus, to map the
5 existing resource to a new memory address, the hypervisor creates a translation table entry to achieve this result. Therefore, as discussed above, a new translation table entry is created rather than allowing the OS image to change an entry or location within the page frame table
10 itself. If, on the other hand, the OS had made a request to map a resource that was not allocated to its partition, the hypervisor would have only returned an error message (step **508**).

With reference now to **Figure 6**, a high level
15 flowchart illustrating an exemplary process within an operating system image for requesting operations to a logically partitioned platform is depicted in accordance with the present invention. The OS image first determines that a system resource needs to be modified to
20 accomplish a task within the OS image (step **602**). The OS image then determines whether the system resource that needs to be modified is one for which access is denied by the logically partitioned platform on which the OS image runs (step **604**). If access is not denied, then the OS
25 image directly modifies the system resource (step **608**). If the system resource is one for which access is denied, then the OS image requests the appropriate service from the hypervisor to achieve the functionally equivalent result of the requested modification (step **606**).

30 It is important to note that while the present invention has been described in the context of a fully

Docket No. AUS990940US1

functioning data processing system, those of ordinary skill in the art will appreciate that the processes of the present invention are capable of being distributed in the form of a computer readable medium of instructions and a variety of forms and that the present invention applies equally regardless of the particular type of signal bearing media actually used to carry out the distribution. Examples of computer readable media include recordable-type media such as a floppy disc, a hard disk drive, a RAM, and CD-ROMs and transmission-type media such as digital and analog communications links.

The description of the present invention has been presented for purposes of illustration and description, but is not intended to be exhaustive or limited to the invention in the form disclosed. Many modifications and variations will be apparent to those of ordinary skill in the art. The embodiment was chosen and described in order to best explain the principles of the invention, the practical application, and to enable others of ordinary skill in the art to understand the invention for various embodiments with various modifications as are suited to the particular use contemplated.

Docket No. AUS990940US1

CLAIMS:

What is claimed is:

1. A logically partitioned data processing system,
5 comprising:
a plurality of logical partitions;
a plurality of operating systems, each assigned to a
separate one of the plurality of logical partitions;
a plurality of assignable resources, wherein each of
10 the plurality of assignable resources is assigned to one
of the plurality of logical partitions;
at least one non-assignable resource; and
a hypervisor, wherein the hypervisor provides a set
of services to each of the plurality of logical
15 partitions allowing a desired result to be achieved by
each operating system without modifying the
non-assignable resource.
2. The logically partitioned data processing system as
20 recited in claim 1, wherein the set of services comprise
a service for creating a new translation table for
mapping a change in a logical address to a physical
address without modifying an existing translation table.
- 25 3. The logically partitioned data processing system as
recited in claim 2, wherein the existing translation
table is a page frame table.
4. The logically partitioned data processing system as
30 recited in claim 1, wherein the non-assignable resource
is a page frame table.

003090" E 3362560

5

5

5

10

15

15

20

25

30

partition resource.

5

10

15

20

25

30

first instructions for receiving, at a hypervisor, a request from an operating system to perform an operation;

second instructions, responsive to a determination that the request would not result in direct access by the operating system to an unassigned resource, for performing the operation.

15. The computer program product as recited in claim 14,
further comprising:

10

15

20

25

30

Docket No. AUS990940US1

second instructions, responsive to a determination that the system resource is one for which access is denied to the operating system, for requesting a service from a hypervisor to accomplish a functionally equivalent task.

19. The computer program product as recited in claim 18, further comprising:

third instructions, responsive to a determination that the system resource is not one for which access is denied to the operating system, for directly accessing the system resource to apply the modification.

20. The computer program product as recited in claim 18, wherein the hypervisor is implemented as firmware.

21. The computer program product as recited in claim 18, wherein the computer program product comprises an operating system.

22. A system for protecting the integrity of a logically partitioned data processing system, the system comprising:

first means for receiving, at a hypervisor, a request from an operating system to perform an operation;

second means, responsive to a determination that the request would not result in direct access by the operating system to an unassigned resource, for performing the operation.

23. The system as recited in claim 22, further

Docket No. AUS990940US1

comprising:

third means, responsive to a determination that the request would result in direct access by the operating system to an unassigned resource, for refraining from performing the operation.

24. The system as recited in claim 22, wherein the request is a request to map a partition resource to a memory address and performing the operation comprises creating a translation table entry to map the memory address to an entry in a page frame table, wherein the entry in the page frame table corresponds to the partition resource.

25. The system as recited in claim 22, wherein the hypervisor is implemented as firmware.

26. A system for providing for modification of system resources by an operating system within a logically partitioned data processing system, the system comprising:

first means for determining that a system resource needs to be modified;

second means, responsive to a determination that the system resource is one for which access is denied to the operating system, for requesting a service from a hypervisor to accomplish a functionally equivalent task.

27. The system as recited in claim 26, further comprising:

third means, responsive to a determination that the

00555555 00555555 00555555 00555555

system resource is not one for which access is denied to the operating system, for directly accessing the system resource to apply the modification.

- 5 28. The system as recited in claim 26, wherein the
hypervisor is implemented as firmware.

Docket No. AUS990940US1

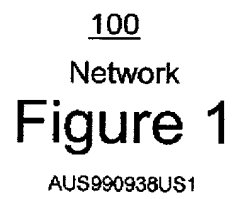
ABSTRACT OF THE DISCLOSURE

HYPERVISOR AS A SET OF SERVICES

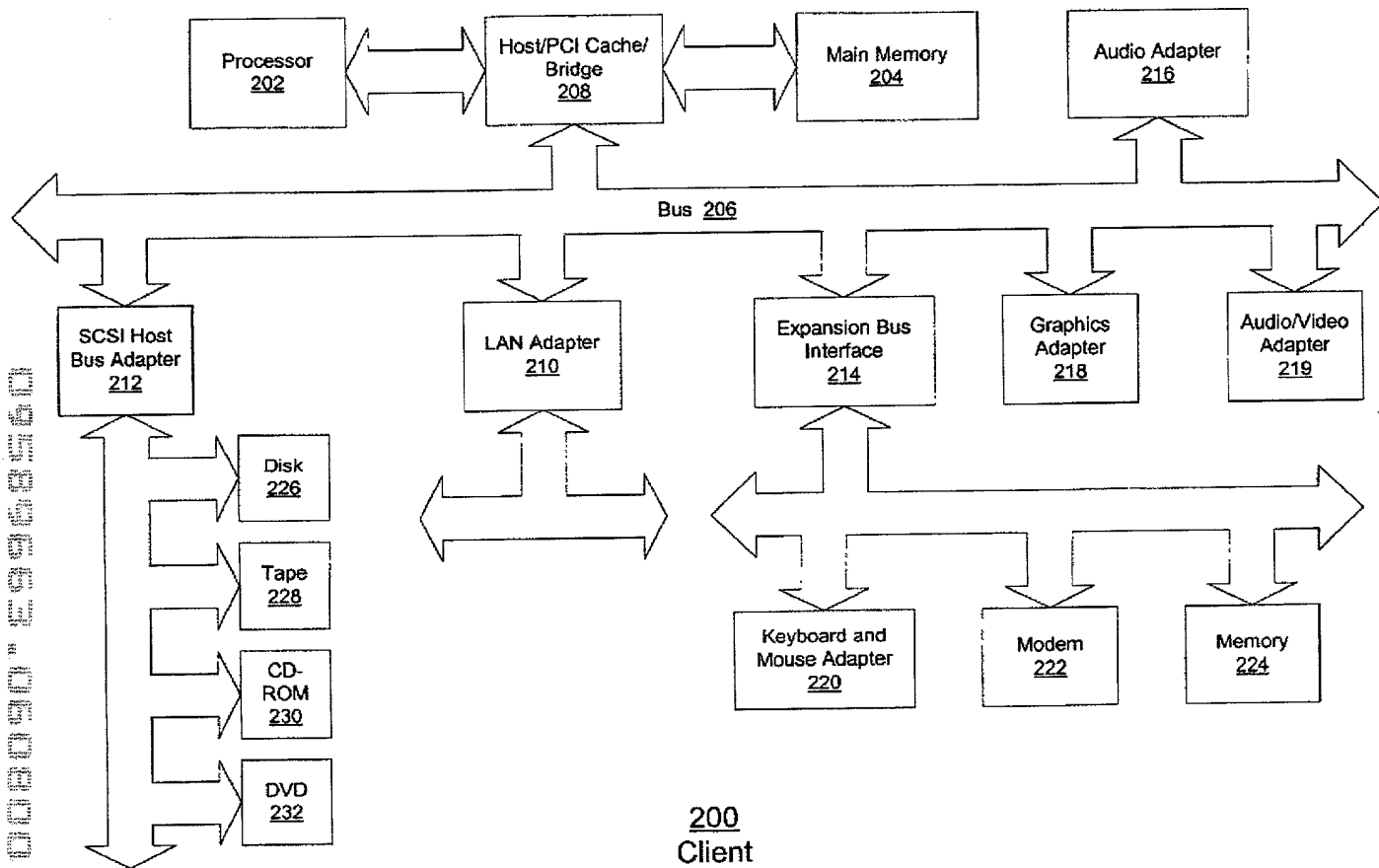
5

A method, apparatus, and system for preventing each of a plurality of operating system within a logically partitioned data processing system from interfering with the operation of the other operating systems is provided.

10 In one embodiment, a logically partitioned data processing system includes a plurality of logical partitions; a plurality of operating systems, a plurality of assignable resources, at least one non-assignable resource, and a hypervisor. Each of the plurality of
15 operating systems is assigned to a separate one of the plurality of logical partitions and each of the plurality of assignable resources is assigned to one of the plurality of logical partitions. The hypervisor provides a set of services to each of the plurality of logical
20 partitions, wherein these services safely perform modifications to non-assignable processing system resources in response to operating system requests without allowing direct access to the non-assignable resources by the operating system image. Thus, each
25 operating system is prevented from modifying the non-assignable resource in such a way that interferes with the operation of other ones of the plurality of operating systems.



AUS990938US1



200
Client
Figure 2

AUS990938US1

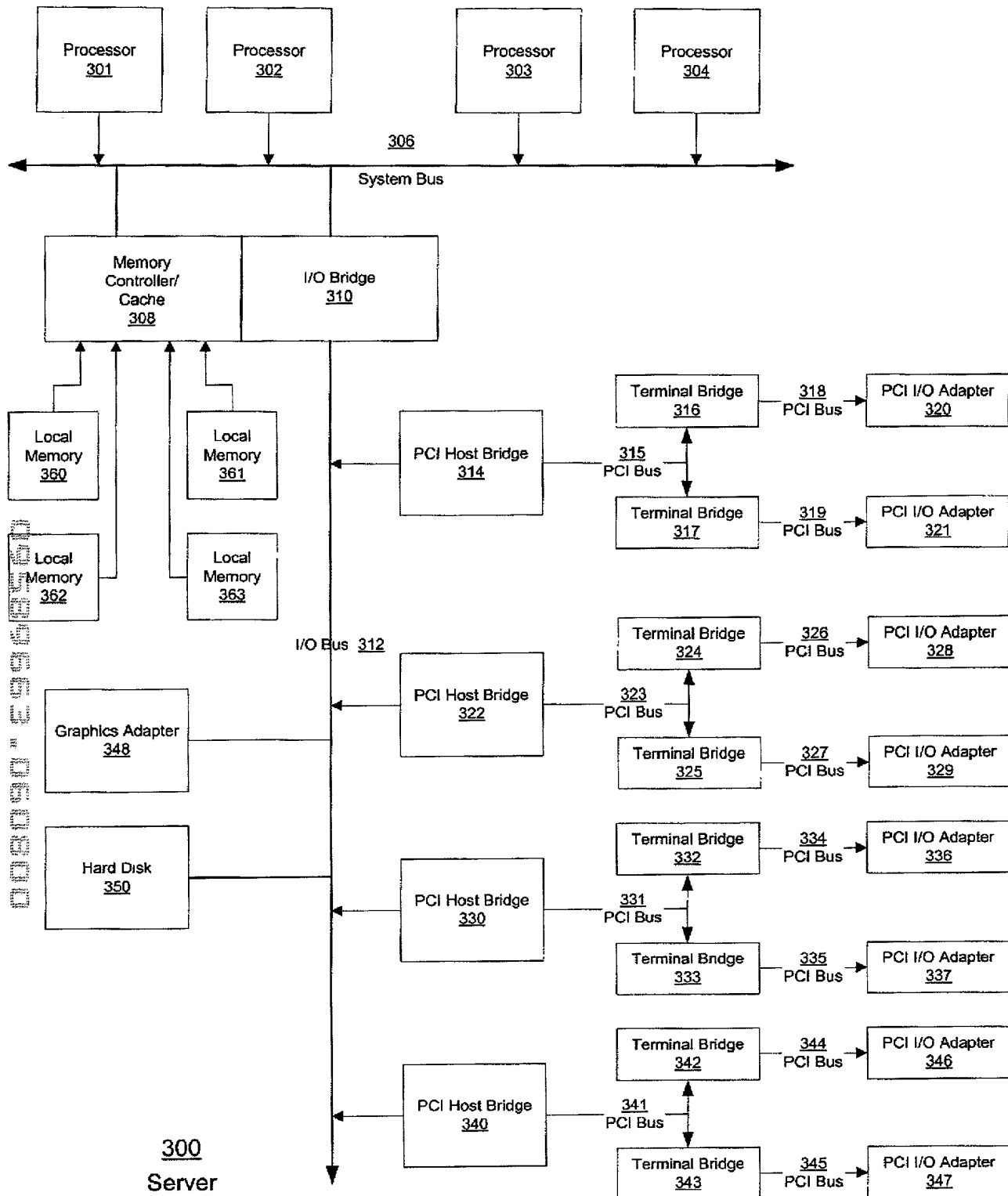


Figure 3

AUS990938US1

00990938US1

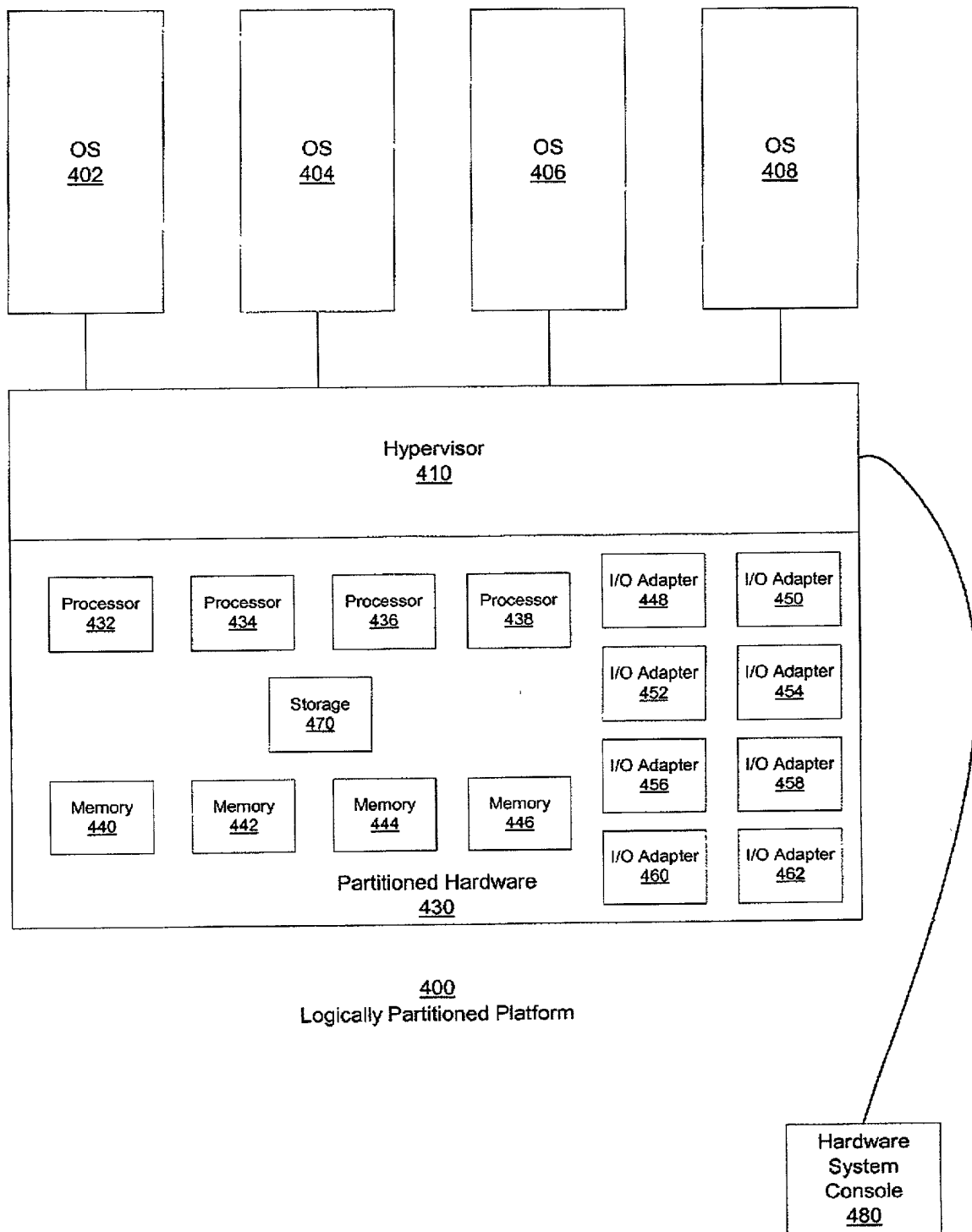
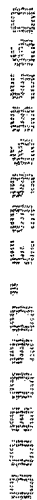


Figure 4

AUS990938US1

Abstract The purpose of this study was to determine the effect of a 12-week training program on the heart rate (HR) and heart rate reserve (HRR) of sedentary, middle-aged men. The subjects were divided into two groups: a control group and an exercise group. The control group consisted of 10 men who did not exercise regularly, and the exercise group consisted of 10 men who exercised regularly. The exercise group performed a 12-week training program consisting of three sessions per week, each lasting 30 minutes. The training program was designed to increase the HR and HRR of the exercise group. The HR and HRR were measured at the beginning and end of the 12-week training program. The results showed that the exercise group had a significantly higher HR and HRR at the end of the 12-week training program compared to the control group. The HR of the exercise group increased from 145 to 155 beats per minute, and the HRR increased from 35 to 45 beats per minute. The HR of the control group remained at 145 beats per minute, and the HRR remained at 35 beats per minute. The results of this study suggest that a 12-week training program can effectively increase the HR and HRR of sedentary, middle-aged men.



Abstract The purpose of this study was to determine the effect of a 12-week training program on the heart rate (HR) and heart rate reserve (HRR) of sedentary, middle-aged men. The subjects were divided into two groups: a control group and an exercise group. The control group consisted of 10 men who did not exercise regularly, and the exercise group consisted of 10 men who exercised regularly. The exercise group performed a 12-week training program consisting of three sessions per week, each lasting 30 minutes. The training program was designed to increase the HR and HRR of the exercise group. The HR and HRR were measured at the beginning and end of the 12-week training program. The results showed that the exercise group had a significantly higher HR and HRR at the end of the 12-week training program compared to the control group. The HR of the exercise group increased from 145 to 155 beats per minute, and the HRR increased from 35 to 45 beats per minute. The HR of the control group remained at 145 beats per minute, and the HRR remained at 35 beats per minute. The results of this study suggest that a 12-week training program can effectively increase the HR and HRR of sedentary, middle-aged men.

Abstract The purpose of this study was to determine the effect of a 12-week training program on the heart rate (HR) and heart rate reserve (HRR) of sedentary, middle-aged men. The subjects were divided into two groups: a control group and an exercise group. The control group consisted of 10 men who did not exercise regularly, and the exercise group consisted of 10 men who exercised regularly. The exercise group performed a 12-week training program consisting of three sessions per week, each lasting 30 minutes. The training program was designed to increase the HR and HRR of the exercise group. The HR and HRR were measured at the beginning and end of the 12-week training program. The results showed that the exercise group had a significantly higher HR and HRR at the end of the 12-week training program compared to the control group. The HR of the exercise group increased from 145 to 155 beats per minute, and the HRR increased from 35 to 45 beats per minute. The HR of the control group remained at 145 beats per minute, and the HRR remained at 35 beats per minute. The results of this study suggest that a 12-week training program can effectively increase the HR and HRR of sedentary, middle-aged men.

```
graph TD; Start([Start]) --> 602[Determine that a system resource needs to be modified. 602]; 602 --> 604{Is the system resource needed one for which access is denied? 604}; 604 -- YES --> 606[Request appropriate service from hypervisor to accomplish task. 606]; 606 --> Stop([Stop]); 604 -- NO --> 608[Make request modification to the system resource. 608]; 608 --> Stop;
```

Figure 6

AUS990940US1

**DECLARATION AND POWER OF ATTORNEY FOR
PATENT APPLICATION**

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name;

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled

HYPERVERSOR AS A SET OF SERVICES

the specification of which (check one)

X is attached hereto.

— was filed on _____
as Application Serial No. _____
and was amended on _____
(if applicable)

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to the patentability of this application in accordance with Title 37, Code of Federal Regulations, §1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, §119 of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

Prior Foreign Application(s):

Priority Claimed

____ Yes ____ No
____ (Number) ____ (Country) ____ (Day/Month/Year)

I hereby claim the benefit under Title 35, United States Code, §120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, §112, I acknowledge the duty to disclose information material to the patentability of this application as defined in Title 37, Code of Federal Regulations, §1.56 which occurred between the filing date of the prior application and the national or PCT international filing date of this application:

____ (Application Serial #) ____ (Filing Date) ____ (Status)

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

POWER OF ATTORNEY: As a named inventor, I hereby appoint the following attorneys and/or agents to prosecute this application and transact all business in the Patent and Trademark Office connected therewith.

John W. Henderson, Jr., Reg. No. 26,907; Thomas E. Tyson, Reg. No. 28,543; James H. Barksdale, Jr., Reg. No. 24,091; Casimer K. Salys, Reg. No. 28,900; Robert M. Carwell, Reg. No. 28,499; Douglas H. Lefevre, Reg. No. 26,193; Jeffrey S. LaBaw, Reg. No. 31,633; David A. Mims, Jr., Reg. 32,708; Volel Emile, Reg. No. 39,969; Anthony V. England, Reg. No. 35,129; Leslie A. Van Leeuwen, Reg. No. 42,196; Christopher A. Hughes, Reg. No. 26,914; Edward A. Pennington, Reg. No. 32,588; John E. Hoel, Reg. No. 26,279; Joseph C. Redmond, Jr., Reg. No. 18,753; Marilyn S. Dawkins, Reg. No. 31,140; Mark E. McBurney, Reg. No. 33,114; Duke W. Yee, Reg. No. 34,285; Colin P. Cahoon, Reg. No. 38,836; Joseph R. Burwell, Reg. No. 44,468; Rudolph J. Buchel, Reg. No. 43,448; and Stephen R. Loe, Reg. No. 43,757, Stephen J. Walder, Reg. No. 41,534.

Send correspondence to: Duke W. Yee, Carstens, Yee & Cahoon, LLP, P.O. Box 802334, Dallas, Texas 75380 and direct all telephone calls to Duke W. Yee, (972) 367-2001

FULL NAME OF SOLE OR FIRST INVENTOR: RICHARD LOUIS ARNDT

INVENTORS SIGNATURE: Richard Louis Arndt DATE: 1 June 2000

RESIDENCE: 1607 BARN SWALLOW DRIVE
AUSTIN, TEXAS 78746

CITIZENSHIP: UNITED STATES

POST OFFICE ADDRESS: SAME AS ABOVE